An Anonymous, Verifiable Internet Service POLL system

Andreas Put *Italo Dacosta* *Milica Milutinovic*
Bart De Decker

Report CW 669, July 2014

Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

An Anonymous, Verifiable Internet Service POLL system

Andreas Put *Italo Dacosta* *Milica Milutinovic*
Bart De Decker

Report CW 669, July 2014

Department of Computer Science, K.U.Leuven

Abstract

While there is a great number of electronic voting protocols proposed in the literature, only a handful of them are actually deployed. Still, these few available schemes require voters to completely trust the poll provider with the anonymity of their votes and/or the integrity of the results. More robust schemes with better privacy and integrity guarantees do exist, however, they are complex to deploy and, therefore, are not suitable for small to medium scale voting scenarios (e.g. electing the board of directors of an international society). In this paper, we present avisPoll, a practical electronic voting scheme that provides flexible anonymity as well as universal poll integrity validation. For this purpose, avisPoll relies on an anonymous credential system and other cryptographic building blocks. The system can be offered as a cloud service which can give everybody the possibility to organize a poll and define the eligible voter set. Hence, the complexity of setting up a poll is significantly reduced without reducing privacy and integrity guarantees. Finally, we show a prototype implementation of avisPoll and give detailed performance results. These results demonstrate that the system is indeed practical for use, even on commodity hardware. For instance, it takes $\approx$ 150 milliseconds for a client to cast a vote, and $\approx$ 25 seconds for server to thoroughly verify 1000 votes.

An Anonymous, Verifiable Internet Service POLL system

Andreas Put, Italo Dacosta, Milica Milutinovic, Bart De Decker
KU Leuven, Dept. of Computer Science, iMinds-DistriNet
Celestijnenlaan 200A, 3001 Heverlee, Belgium
firstname.lastname@cs.kuleuven.be

ABSTRACT

While there is a great number of electronic voting protocols proposed in the literature, only a handful of them are actually deployed. Still, these few available schemes require voters to completely trust the poll provider with the anonymity of their votes and/or the integrity of the results. More robust schemes with better privacy and integrity guarantees do exist, however, they are complex to deploy and, therefore, are not suitable for small to medium scale voting scenarios (e.g. electing the board of directors of an international society). In this paper, we present avisPoll, a practical electronic voting scheme that provides flexible anonymity as well as universal poll integrity validation. For this purpose, avisPoll relies on an anonymous credential system and other cryptographic building blocks. The system can be offered as a cloud service which can give everybody the possibility to organize a poll and define the eligible voter set. Hence, the complexity of setting up a poll is significantly reduced without reducing privacy and integrity guarantees. Finally, we show a prototype implementation of avisPoll and give detailed performance results. These results demonstrate that the system is indeed practical for use, even on commodity hardware. For instance, it takes $\approx$ 150 milliseconds for a client to cast a vote, and $\approx$ 25 seconds for server to thoroughly verify 1000 votes.

1. INTRODUCTION

Opinion polls, surveys, referenda and elections are all instruments to solicit and collect the people's opinions, convictions or preferences. In the past, casting a vote or taking part in a poll required physical presence of the voter. When the majority of population gained access to telephones, polls could be taken remotely over the telephone, facilitating people to express their opinion. Today, we can observe a trend towards electronic and Internet voting. Internet voting further facilitates the voting process, allowing voters to cast their vote at any time during the poll, from virtually anywhere. In addition, due to its electronic nature, Internet voting allows for fast, automated tallying of the results. Furthermore, it has severely lowered the requirements to set up a poll. Websites and fora often organize opinion polls to discover the interests of their users. There even exist dedicated service providers that offer to organize a poll in only a few easy steps [16, 9].

However, Internet voting has several severe drawbacks, which might not be apparent at first sight. Anonymity and unlinkability of the voter and her vote is important in order to avoid coercion or pressure by peers or lobbies. This is not an easy requirement to fulfil without leaving a digital trace of the voter's identity, as only legitimate voters should be able to vote and nobody should be able to vote twice. In addition, because people can participate in an Internet poll from virtually anywhere at virtually any time, the risk of coercion increases. Because of the electronic nature of Internet voting, the voting server can also tamper with the vote outcome in an automated fashion. Another difficulty for many Internet-based polls is to prevent double voting. Often, voters are first required to register an account with the poll provider, usually recording only their email address. This weak requirement allows malicious voters to easily set up multiple identities with the poll provider in order to influence the poll results.

Any failure to fulfil the anonymity and integrity requirements by a voting system is a cause to question the poll results it produces. If a voter can be linked to her vote, she might decide not to participate in the poll, or express her own opinion. Moreover, a coercer could verify in which way someone voted and voters could easily sell their votes since they would be able to prove that they voted a certain way. In addition, if the poll result could be influenced without it being detectable, the final outcome can never be trusted. This is certainly the case for polls where the poll server has an interest in the poll results. However, even if the poll server is honest, it could be attacked by a third party or infected by malware, allowing the poll results to be tampered with [13].

In general, three types of Internet-voting systems can be identified: 1) systems where the voter has to trust the server to provide anonymity *and* integrity, 2) systems where the voter has to trust the server to provide anonymity *or* integrity and 3) systems that do not need to be trusted to provide any guarantees. Of course, the latter type of system is the most desirable. However, these systems often require an elaborate setup and use complex and time-consuming cryptographic tools, such as mix-nets and homomorphic encryption. The second type does not require an elaborate setup

WOODSTOCK '97 El Paso, Texas USA

and some are currently even offering their services over the internet [17]. Finally, current website/fora-based polls fall into the first category, where the voter is required to fully trust the system. These are only suitable for polls where very little is at stake, as there are many examples of poll organizers and attackers tampering with the outcome [11, 15].

In this paper we present a new electronic poll system, avisPoll, which offers flexible anonymity, significantly reduces the setup requirements and needs less trust in the different system entities. Moreover, the integrity of the poll results is universally verifiable. Hence, avisPoll tries to find the middle ground between the second and third type of systems described above. The trust requirements of avisPoll allow for a poll service provider to offer the service of gathering and processing polls over the Internet. Anyone can organize an online poll with this provider without requiring it to provide voter anonymity or integrity guarantees. This system is mainly designed with *low-risk elections* in mind, i.e. elections where the stakes are not too high, or the risk for coercion on a large scale is non-existent (e.g. electing the representatives on a board, or choosing the chairman of a meeting).

Voters first need to obtain a avisPoll account in order to obtain an anonymous voting credential which can be used to vote. The unlinkability properties allow avisPoll voters to use the same credential in multiple polls while still preventing voters to cast multiple votes in the same poll. In addition, the anonymous credential's selective disclosure properties allow avisPoll organizers to target a specific group of voters (e.g. adult females) without breaking the system's anonymity properties.

This paper makes the following key contributions:

- We present avisPoll, an electronic voting scheme that provides flexible anonymity as well as universal poll integrity validation. avisPoll can be offered by an online provider, which can allow any entity, whether it is a large organization or the representative of a small group of people, to set up a poll. Voters are required to register in order to obtain an anonymous voting credential. However, the same credential can be used in multiple transactions thanks to the properties of anonymous credentials. Furthermore, avisPoll requires only little trust in the poll service provider and the poll organizer, and can be extended to even lessen the required trust.

- We made a prototype implementation of avisPoll and show extensive benchmarks. In doing so, we prove that avisPoll is scalable and practical to use, even on commodity hardware.

The paper is organized as follows: Section 2 gives an overview of the related work and Section 3 briefly explains the preliminaries. Section 4 explains the scenario for avisPoll in more detail, as well as our threat model. avisPoll's protocols are listed in section 5. We benchmark our prototype implementation in section 6 after which avisPoll's properties are discussed. Finally, section 8 contains our conclusion, in which we also show our plans for future work.

2. RELATED WORK

Proposals exist that offer verifiable, i.e. open-audit voting systems. The work of Adida [1] aims to provide strong integrity guarantees, however, vote privacy and unlinkability relies on the fact that a subset of trustees will remain honest. The system proposed by Clarkson et at. [7] is able to offer either vote integrity or vote anonymity, but not both.

The use of anonymous credentials in voting schemes has already been proposed in the past. These credentials are ideal for proving a subset of personal attributes, while hiding the user's identity. Anonymous credentials are already deployed in a real-world pilots. ABC4Trust's pilot program allowed university students to anonymously participate in an online course evaluation [14]. It demonstrates that this technology can be realistically utilised to provide required integrity guarantees and user anonymity.

Proposals like [2, 3, 8] also utilise anonymous credentials. However the entity that processes the votes must be trusted. The user votes are not encrypted, and the integrity is not guaranteed. In addition, users are only allowed to cast a vote once and cannot change their opinion at a later time.

The work of Cetinkaya and Doganaksoy [6] formulates a solution which utilises anonymous credentials to hide users' identities. These credentials represent election information together with a random number, blindly singed by the voting authority. They are unblinded and used as signed authorisations to vote. The voting authority cannot identify the voters, however, with this approach it is impossible to verify the eligibility of users or have targeted elections. On the other hand, a scheme proposed by Joaquim et al. [12] relies on an 'Anonymizer', through which users submit their votes, in order not to be identified by the counting entity. However, the users are not assumed to hide their identity from the anonymising party and targeting voter sets is not considered.

3. PRELIMINARIES

3.1 Anonymous Credentials

Anonymous credentials are cryptographic tokens with which credential owners can prove statements about themselves and relationships with organizations anonymously. avisPoll makes use of the Idemix anonymous credential system [5]. In a typical transaction, Idemix credential owners prove ownership of their credential, in which they can choose to hide certain attributes. Furthermore, users can prove certain facts about their credential by proving additional predicates. The predicates that are important for avisPoll application are: *domain pseudonym* and *range proof*. A domain pseudonym is a value generated based the credential's secret and a domain specification. Every credential owner can generate exactly one unique domain pseudonym per domain. Hence, this predicate can be used to limit the credential usage. By proving a range proof, the credential owner proofs inequalities of attributes in her credential. For instance, she can prove that she is older than eighteen years without revealing her actual birthday. Finally, avisPoll makes use of Idemix message signatures, in which messages are signed using the Fiat-Shamir heuristic [10]

Service providers can specify what exactly users need to proof in *eligibility policies*, such as the ABC4Trust presentation policies [4]. For example, the policy could specify that a user needs to prove ownership of her poll-credential, her

domain pseudonym using the domain 'pollID' and that she is older than eighteen. The user can inspect this policy before she authenticates, and the attributes in her credential which do not have to be revealed by the eligibility policy can be hidden.

3.2 Merkle Trees

A Merkle hash tree is an, often binary, tree where the leafs are the hashed values of data-blocks. The value of a node is computed by hashing the concatenated values of a node's child-node. Using such a tree, the integrity of a data-block can be validated simply by finding the corresponding hashed value in the tree, and validating the leaf's parent nodes. Note that only a part of the tree is required to do this. To validate the authenticity of a data block, the root of the tree needs to be signed and the integrity of the data block needs to be verified together with the signature of the tree-root's signature. Hence, only one signature is required to proof the authenticity of any data block in the tree.

4. avisPoll

4.1 Scenarios

In a simple setup, avisPoll involves three participants: a set of voters, an organization that wants to conduct an electronic poll and a poll provider. The latter is a third-party that collects, processes and maintains the votes in a bulletin board in the cloud. The poll provider offers flexible scalability: supporting votes with only a few participants to polls with thousands, tens of thousands or even more eligible voters. An example of this setting could be polls or plebiscites organized by the authorities (at the level of a city, region or even country) to consult the population about a particular issue as happens quite often in some countries such as Switzerland. By moving this process to the cloud, avisPoll allows organizations to save costs by not having to deploy the infrastructure themselves.

However, with this approach the organization loses a certain degree of control over the poll. The challenge is to guarantee the anonymity of the voters and the integrity of the results (published on a bulletin board) if the poll provider is malicious or is compromised by an adversary. Thus, the main priority of avisPoll is to provide strong guarantees that the poll provider cannot manipulate the poll results without being detected. For example, the poll provider should not be able to learn the contents of the votes before the organization closes the poll. Furthermore, if the outcome of the poll can be known only by the organization, then the tallying of the votes could be performed by the organization itself. For highly sensitive polls, an organization could even deploy their own private avisPoll service to avoid trusting a third-party (at the associated cost).

The responsibility of the organization is to define the poll and the voter's eligibility requirements (i.e. how users prove they are entitled to vote). In this simple scenario, some trust is required by the voter in the organization. The organization can either issue these credentials to the voters itself, or rely on a trusted credential issuer, like the government, to do so. In the latter case, the poll organizer requires less trust from the voters, as it is not in control of the credential issuing process.

4.2 Requirements

Based on the scenario described in the previous section, avisPoll should satisfy the following requirements:

(a) *Vote privacy.* Voting should be anonymous, i.e. it should be computationally difficult for any entity in the system to identify voters by their votes only. Moreover, the data included in each vote should be generic enough to avoid breaking privacy guarantees.

(b) *Vote confidentiality.* The data included in each vote should not be revealed to any entity in the system before the poll is closed, i.e. casted votes should be encrypted. Thus, it should not be possible to discriminate votes based on their content (e.g. omitting a vote if its content is unfavorable). This requirement also implies that it should be computationally hard to tally the votes before the poll is closed, as a partial result might influence the eventual outcome.

(c) *Vote unlinkability.* Different votes (in different polls) of the same voter should not be linkable, in order to ensure both forward and backward secrecy of the voting process.

(d) *Vote unforgeability.* It should be computationally hard for an adversary to forge votes without being detected.

(e) *Individual verifiability.* A voter can verify that her own vote is included in the published poll results and that it is correct (i.e. it has not been modified).

(f) *Universal verifiability.* Any entity, internal or external to the system, should be able to verify the correctness of any vote in the published results. In particular, anyone should be able to check that each vote in the published results was cast by a valid voter, the vote data has not been modified and only one vote per voter exists.

(g) *Double-voting prevention.* Each eligible voter can only cast one valid vote. The system could allow voters to cast multiple votes, in which case only the last submitted vote will count. In addition, the system could optionally allow voters to withdraw (i.e. cancel) submitted votes before the poll closes.

(h) *Poll integrity.* The system should detect if any legitimate vote is removed from the poll before or after the poll is closed (hence, when the collector of the votes is unreliable, it can be proven that the poll was manipulated).

(i) *Prevention of large-scale coercion.* The system should not allow for large scale coercion or vote selling (i.e. voters should not receive a receipt with which they can prove how they voted). For this purpose, the system may also allow voters to update the votes before the poll closes.

(j) *Practical trust model.* The system should only require a small number of entities with limited trust.

(k) *Voter selection.* The system should allow to precisely target a particular set of eligible voters (e.g. only female students of MIT or CMU universities in the age between 20 to 22 are allowed to participate).

(l) *Selective disclosure.* The system should allow for collecting extra information about the voters; this extra information should be correct (i.e. endorsed by a trusted party). Moreover, the collection of additional information about a voter should always be subject to the voter's consent.

(m) *Scalability.* The system should be efficient and scalable, i.e. it should support polls with tens of voters to polls with thousands or more voters.

4.3 The avisPoll roles

Poll Service Provider (PSP).

The PSP offers poll services to organizations for a fee. It allows organizations to create and configure polls. For each poll, the PSP receives, validates, processes and stores the votes cast by eligible voters. Moreover, the PSP vouches for the integrity of the bulletin board that contains the stored votes. If a organization chooses to, the PSP could also tally the votes once the poll is closed. In most scenarios, other entities have no access to the votes processed by the PSP (i.e. votes are stored in servers controlled only by the PSP). Furthermore, the PSP may also provide the client application that is used to participate in the poll. Still, organizations could also choose client applications internally developed or from other providers (we assume that the voting protocols are publicly known).

Poll Organizer (PO).

The PO represents the organization that wants to create a new poll using PSP's services. When creating a new poll, the PO defines different properties such as: questions, eligibility policy, validity period and participant entities. Using the eligibility policy, the PO defines the type of credentials and attributes voters need to have in order to participate in the poll.

In small scenarios, it is likely that the PO will also issue credentials and receive complaints from voters. However, if voters cannot trust the PO with the outcome of the poll, additional entities may be required to reduce the risk of such a threat, as we describe next.

Poll Auditor (PA).

The PA's goal is to reduce the risk of fraud by the PO (e.g. if the PO colludes with the PSP). For this purpose, it validates and certifies the polls created by the PO and the corresponding results. It also splits with the PO any secrets used to encrypt the votes. Moreover, the PA could also monitor that the poll credentials are correctly issued. Also, voters and external reviewers can complain with the PA if any anomaly in the poll process is detected. Depending on the scale and value of the poll, one or more PAs could be used.

Credential Issuer (CI).

The CI issues the credentials that voters need to participate in the poll. It can be part of the organization (e.g. HR department) or a trusted third-party (e.g. the government). In general, credentials are issued following well-defined and audited procedures, e.g. specified by the PA. The use of a CI also reduces the required trust in the PO.

Voter.

A voter is a member of the organization with a valid credential, obtained from the PO or CI. An eligible voter must be able to prove that she owns a valid credential, and that the information contained in this credential can satisfy the eligibility policy.

4.4 Threat model

avisPoll targets poll scenarios with low to medium value and risk of coercion. That is, we do not target high risk polls or elections, where some parties might want to buy votes from voters or coerce them to vote in a particular way. Thus, avisPoll should not be used when the stakes are high.

In our threat model, organizations have low trust in the PSP. The PSP could be malicious or it could be compromised by an adversary. A *malicious or compromised PSP* may try, therefore, to change the outcome of the poll. For example, the PSP may try to manipulate votes, inject forged votes, deny or delete genuine votes, report wrong results, or replace votes with older versions of themselves (if the poll allows voters to update their votes). Moreover, a malicious or compromised PSP may try to link votes to the identities of the voters or reveal the results of the poll to non-authorized parties, thus, breaking the anonymity and confidentiality guarantees of our system.

Voters could also mistrust the PO. For example, a *malicious PO* may collude with the PSP to manipulate the results of the poll (e.g. omit unfavorable votes from the final tally) or to reveal the identity of certain voters. A malicious PO may also try to created specially crafted polls to reveal voters' identities, issue bogus credentials (if the PO has such capability) or share encryption keys with the PSP to reveal the contents of the votes and compute results before the poll is closed. To reduce the impact of a malicious PO, the organization can introduce a separate CI and one or more PAs. In our threat model, the *CI and the PAs are honest-but-curious parties*, i.e. they follow the protocols but the may try to break privacy guarantees based on the data they have or to share such data with other parties (e.g. the PSP).

Voters could also be malicious. During the voting process, *malicious voters* may try to vote even if they are not eligible for the targeted poll (i.e. they do not belong to the target voter set). Malicious voters may also try to vote more than once when such action is not allowed or submit an invalid vote (e.g. random bits). Finally, malicious voters may try to frame the PSP or PO and accuse them falsely of having manipulated the poll.

Finally, our threat model does not consider denial of service attacks or attacks against the voters' computers HW and SW, including the poll client application (e.g. large-scale attacks to steal voters' credentials or control voters' computers).

5. AVISPOLL PROTOCOL DESCRIPTION

5.1 Assumptions

We assume that voter anonymity and unlinkability are not affected by other channels (e.g. IP-based tracking). In addition, we assume that the voters have access to the general parameters, keys and certificates made available by the PO and PSP. The avisPoll client has access to a local copy of the PSP's and PO's certificate authority. Finally, the following

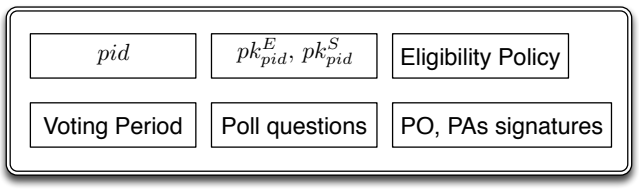

Figure 1: Poll Description contents

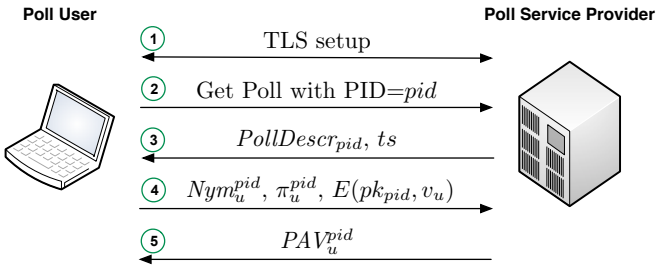

Figure 2: avisPollvoting protocol. After a TLS connection has been set up and the client has authenticated the server, the client sends an Idemix proof in which the encrypted vote is signed.

protocols describe avisPoll with all its entities, i.e. including PAs and CI. Note that the responsibilities of these two entities can be taken up by the PO.

5.2 Voter registration

Before a voter can cast a vote, she needs to obtain an anonymous Idemix credential, referred to as the avisPoll-credential. These credentials are issued by the PO or a trusted credential issuer (e.g. the government). avisPoll-credentials can contain additional, personal information which can be used by the PO to limit the eligible voter set. Note that using personal information in an anonymous credential is not privacy-intrusive (see section 3.1). The PO decides when voters are able to register for a credential, e.g. it can allow or deny voters to register when a poll is already started.

In order to obtain such a credential, the voter first authenticates, after which her identity is validated. Finally, the voter is issued a corresponding avisPoll-credential. This process can be audited by a PA.

5.3 Poll setup

To set up a poll with the PSP, the PO generates a *poll description* (figure 1).

This description contains a unique poll identifier, the questions of the poll and the eligibility policy which is used to select the target voter group and avisPoll-credential requirements. Furthermore, the poll description entails the voting period and the public key used to encrypt the votes by the voter. Finally, all this data is signed by the PO (and optionally the PAs), and the resulting signatures are appended to the poll description. A PA formally approves the poll by signing the poll description. Furthermore, users are able to report any detected discrepancies to any PA that signed this description.

As mentioned, the poll description contains a poll public key. This key will be used by voters to encrypt their vote. The corresponding private key is available only to the PO during the poll. However, if a PA is participating, the private key is split among multiple parties. This reduces the risk that the PSP could get hold of this private key and start decrypting votes before the poll is over.

After the poll description is created, it is made public by the PSP, PO and PAs. Finally, the PSP prepares the database in which it will store the valid votes.

5.4 Voting phase

5.4.1 Poll opening

Before the voting phase can start, the PSP prepares the database that will contain the processed votes (table 1). The exact purposes of each table column will be discussed in detail in this section. To signal the start of the voting phase, the PSP inserts a special *opening vote* in the database. This record has a default value as domain pseudonym (OPEN), and does not contain actual vote-related values. However, it does specify a time stamp, which symbolizes the starting time of the poll.

After the voting phase is finished, the PSP inserts a *closing vote* in the database. Similar to the opening vote, this vote specifies CLOSE as domain pseudonym and its time stamp symbolizes the closing time of the poll. The closing vote does not contain actual vote-related values.

5.4.2 Vote creation (voter)

Before a voter can vote, she has to obtain the poll description. First, the signatures contained in the poll description are validated using the voting-client's locally stored certificates. Second, the voting client presents the user with the questions she needs to answer. Thereafter, the vote can be cast according the the protocol described in figure 2.

The avisPoll client app sets up a TLS connection with the PSP. Furthermore, the application authenticates the PSP using its locally stored certificates. Thereafter, the PSP server will send the poll description, $PollDescr_{pid}$ and a time stamp, ts to the avisPoll client. The client app verifies whether this time stamp makes sense (should be close to the current time and greater than any previously received time stamp).

At this point, the client application can construct the client's vote. First, the poll answers are encrypted. The client generates an AES key ($skey$) with which it encrypts the vote and then encrypts $skey$ using the public encryption key (pk^E_{pid}) that is retrieved from the poll description.

$$Vote^E_u \equiv E(pk^E_{pid}, v_u) \equiv asymE(pk^E_{pid}, skey), symE(skey, v_u)$$

Second, it generates the poll-specific domain pseudonym, Nym^{pid}_u, constructs a proof according to the eligibility policy embedded in the poll description, which proves not only the ownership of a valid Idemix credential, but also proves the domain pseudonym and additional predicates if these are required by the eligibility policy. In addition, the encrypted vote and the time stamp are signed in this proof. The proof, π^{pid}_u, together with the encrypted vote and ts is sent to the PSP.

5.4.3 Vote validation (PSP)

When the PSP receives π^{pid}_u, it starts a validation process:

Nym	PIV	PAV	EncryptedVote	Proof	Timestamp	Vote	Symm. key
OPEN	PIV_{open}	PAV_{open}	*null*	*null*	t_{open}	*null*	*null*
CLOSE	PIV_{close}	PAV_{close}	*null*	*null*	t_{close}	*null*	*null*
Nym_u^{pid}	PIV_u	PAV_u	$Vote_u^E$	π_u^{pid}	t_u	$Vote_u$	$skey_u$

Table 1: avisPoll database table structure. In addition, the table lists the contents of the opening and closing votes as well as the contents of a regular vote.

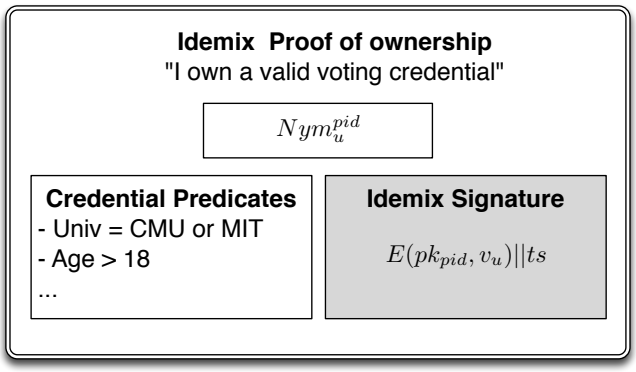

Figure 3: avisPoll vote contents. The voter constructs an Idemix signature-proof of ownership of the avisPoll-credential. In this proof, a domain-pseudonym is proven relative to the poll identifier while the encrypted vote and time stamp are signed. Optionally, further predicates are proven if required by the eligibility policy.

1. Verify that the time stamp used in the signature-proof π_u^{pid} is equal to the one sent to the voter.
2. Check if the domain pseudonym, Nym_u^{pid}, is present in the database.
   (a) Refuse or update the vote (if allowed) if the domain pseudonym is present and the time stamp, ts, is more recent than that of the stored vote, t_u.
3. Check the validity of signature-proof π_u^{pid}.
4. Verify whether π_u^{pid} satisfies the eligibility policy.

If the proof validates, the PSP constructs a *poll integrity value* or PIV_u. This value is a cryptographic hash of the received poll information. ($Vote_u^E$ represents the encrypted vote.)

$$\text{PIV}_u = \mathcal{H}(Nym_u^{pid} \,||\, Vote_u^E \,||\, \pi_u^{pid} \,||\, ts)$$

Next, the PSP creates the *poll authenticity value* or PAV_u. This is a signature on PIV_u using the PSP's private key, pr_{pid}^S, corresponding to the public key, pk_{pid}^S, contained in the poll description.

$$\text{PAV}_u = \mathcal{S}(pr_{pid}^S, \text{PIV}_u)$$

Thereafter, the PSP inserts a new record in the database (or updates an existing record), the values of which can be seen in table 1. The 'vote' and 'symmetric key' values of this record are reserved for the Tally phase and, hence, remain empty.

Finally, the PSP sends PAV_u to the voter, and thereby, the PSP commits to the user that it has processed and registered the voter's vote. *The PSP hereby hands the evidence to the user that his vote is in the database.*

5.4.4 PAV_u validation (voter)

After PAV_u is received, the user computes PIV_u from its local data and uses it to validate the received PAV_u. The avisPoll client app stores PAV_u, π_u^{pid}, ts and the vote (v_u), as it might be required in the rebuttal period. If the PSP does not provide a (correct) PAV, the client application will send a complaint to the PO (and PAs).

5.5 Poll closure

The PSP will insert the CLOSE record in its database when the poll end-date, as specified in the poll description, has passed. Next, the PSP constructs a Merkle-tree from every PIV_u in the database, including PIV_{open} and PIV_{close}. The PSP signs the top node of the tree, and sends the tree, signature and an image of its database to the PO (and PAs). The PO (and PAs) verify the tree, after which the tree is signed again and published.

5.6 Rebuttal period

Once the tree is signed by another entity than the PSP, the latter is not able to tamper with the poll results *without it being universally detectable*. The PSP can indeed remove votes from its database before constructing the tree. However, this can be detected by the owner of the deleted vote, which has the PAV_u as proof that the missing vote was properly recorded. The rebuttal period is a period between the poll closure and the tally phase in which users can report missing votes. If missing votes are detected, the votes can either be recast, or the poll can be declared invalid. Recasting a vote means sending the original vote together with PAV_u to the PO (or PA), who validates the proof and PAV_u before reinserting the vote. Note that the rebuttal period could also be part of the verification phase.

5.7 Tally phase

Before the votes can be tallied, they need to be decrypted. If one or more PAs are participating in the poll, the PO and PAs recombine their key-shares into the private decryption key pr_{pid}^E. For each vote in the database, the encrypted vote is decrypted and the vote's record is updated with the decrypted vote and the symmetric key *skey* with which the vote is encrypted. Once the votes are all decrypted, the tally operation is trivial.

5.8 Verification

At any point after the PSP has released its database containing the votes, the PO, PAs, voters and other entities can

start verifying the integrity of the poll.

5.8.1 Vote verification

To validate a single vote, the following actions need to be performed:

1. Retrieve the database row of the the desired vote. This can be done either at random, or a domain pseudonym can be used as key.
2. Check if the vote's time stamp is between the opening an closing vote's time stamp or the times specified in the poll description.
3. Verify if no other vote has been cast using the vote's domain pseudonym.
4. Verify the Idemix proof as well as the fact that the proof's signature is correct. Also verify that the proof satisfies the poll's eligibility policy. This verifies whether the correct domain pseudonym was generated as well as whether or not the voter was eligible to vote.
5. Compute the PIV_u and verify whether it is equal to the one in the database row.
6. Verify the PAV_u with regard to the computed PIV_u.
7. Verify if the vote has been decrypted correctly by decrypting the encrypted vote using the database row's 'symmetric key' attribute, and match the result with the 'decrypted vote' attribute of the database row.

Note that this process is less complicated if a voter verifies her own vote. She can retrieve her vote using her domain pseudonym and match the PAV_u directly with her local copy. A mismatch would mean that the vote has been tampered with, as the local PAV_u is already validated by the voter when it was received. Hence, steps 2 and 4-6 are not required when a voter validates her own vote.

5.8.2 Validating the Merkle-tree

The Merkle tree can either be partially or fully validated. The full validation requires to check the presence of every PIV_u in the tree, and to check the absence of leaf-nodes which are not a PIV_u in the database. In addition, the other nodes (which are the result of hashing their two child nodes) have to be validated. This whole process is most easily performed by recreating the whole tree, and matching the top nodes of the original and newly created one. Finally, the signatures of the top node have to be validated.

To partially validate a the tree, the presence of each to be verified PIV is checked in the tree. Afterwards, the branches leaving from the PIV's leaf node to the top node are validated as well as the top node's signatures.

6. PRACTICAL EVALUATION

6.1 Performance

We have implemented the avisPoll system as a client and server application in the Java programming language. We installed the avisPoll server component on a workstation with an Intel® Core™ i7-3770 CPU, Ubuntu 13.04 (linux 3.11). The server is equipped with mysql 14.14 and Java SE 1.7.

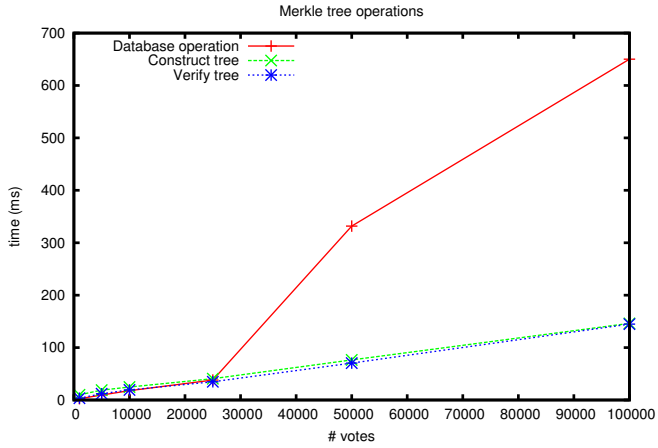

Figure 4: Times in milliseconds of the Merkle tree related operations. We measured the total time for the database operation, tree construction and tree verification.

The client application ran on a machine with an Intel® Core™ i5-3210M and Java SE 1.7.

To cast a vote, the client constructs a proof which satisfies the eligibility policy. In this case, the policy specified that the voter proves ownership of her voting credential, an Idemix credential with seven attributes and a 2048-bit key size, and proves a domain pseudonym relative to the poll identifier. In this proof, she signs her encrypted vote and the received time stamp. Afterwards, the client receives a PAV, which she verifies by reconstructing the PIV and validating the received PAV based on the locally created PIV. This process was repeated 100 times. The proof generation takes on average 142.5 ms (standard-deviation = 8). Verifying the PAV takes on average 4 ms (standard-deviation = 2)[1].

To process the incoming votes, the server needs to validate the Idemix proof. This includes checking whether the proof satisfies the eligibility policy. In addition, the server needs to create the PIV and PAV, and finally create a new record in the database before sending the PAV to the client. The PIV uses SHA-256 as hashing algorithm, while the PAV is created using a 2048-bit RSA signature. The test were repeated 100 times. Proof validation takes on average 105 ms (standard-deviation = 5). Creating the PIV and PAV requires 11 ms (standard-deviation = 4) and finally the database insert takes on average 47 ms (standard-deviation = 17.5)[2]. Except for the database operation processing proofs can easily be parallelized.

To illustrate the scalability we have ran tests on our server machine to measure the Merkle tree operations, vote decryption and vote verification on a database containing 1k, 5k, 10k, 25k, 50k and 100k votes. Figure 4 shows the times to execute the two core operations involved in retrieving the PIVs from the database and constructing the tree. In addition, the time required to verify the tree is listed, during which the tree is reconstructed (we assume that a list of all

[1]All measurements do not include network time.
[2]Note, however, that we have not optimized the database. Hence, this result could be improved.

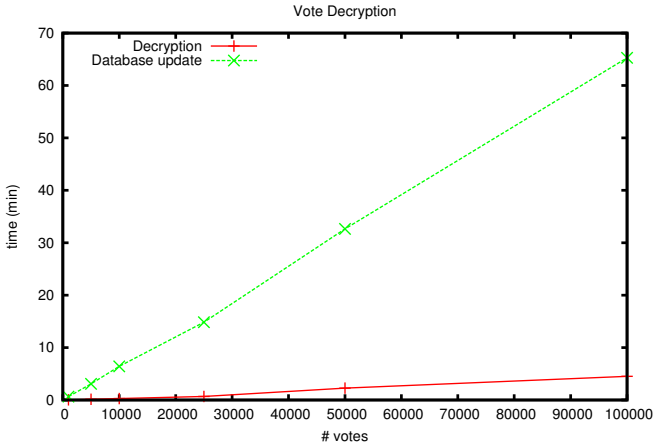

Figure 5: avisPoll decryption time in minutes. We measured the required time for decrypting the encrypted votes as well as the time to update the database.

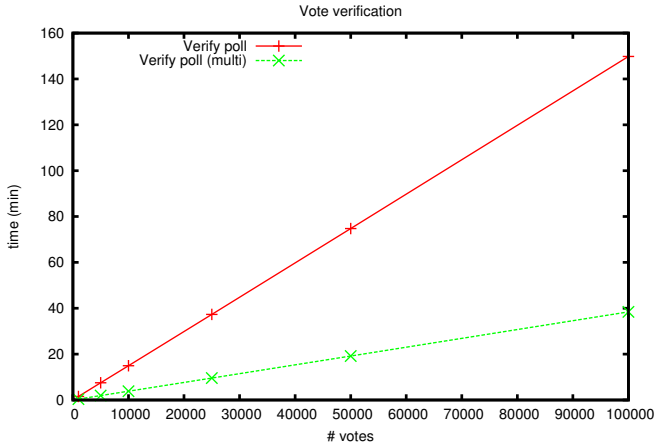

Figure 6: avisPoll total vote verification time in minutes. To illustrate the scalability of avisPoll we compare a single-threaded approach with a multi-threaded one. A performance increase with a factor of nearly four is achieved on a quad-core CPU.

PIVs is present), the top-nodes of the original and reconstructed trees are compared and the signature is verified. The results show that all three operations initially take up about the same time and slowly increase linearly until we reach the 50k database. The times to construct and verify the tree increases at the same rate, however the database operation grows much faster. This is probably due to memory issues caused by the collection of a large amount of objects in a list.

Next, we measured the required time for decrypting the poll and updating the records (figure 5). The first step is the actual vote decryption. This operation is relatively cheap. Our system decrypted one hundred thousand votes in about 4.5 minutes and scales linearly with the amount of votes. The database update, however, is more time consuming. Updating five thousand records already takes longer than decrypting 50 000 votes. Where the vote decryption grows with about 2.7 ms per vote, the update operation grows with nearly 40 ms per operation. Again, this can be improved by optimizing the database.

The results of vote verification can be seen in figure 6. As expected, the poll verification increases linearly with the number of votes in the database. Verifying 10 000 votes takes up almost 15 minutes, while verifying 100 000 votes costs nearly 150 minutes on our server. Since verifying a large poll should not be an interactive process, we consider this a good result. Furthermore, verifying 1000 votes is done within 25 seconds. By means of extrapolation, we can infer that verifying 100 votes would take less than 2.5 seconds, which is a good result for a small poll.

As can be seen in table 2, the main part of verifying a vote is to verify the Idemix proof, and check if the proof satisfies the eligibility policy. Note that this time might increase with regard to the used credential and required additional predicates in the proof, and, hence, the overall time to verify a vote is affected by the number of attributes in the credential (less is better), and the set of predicates that need to be verified in the proof.

However, even though the system's performance might decrease because of the type of proof, we also show that verifying votes is perfectly parallelizable (figure 6). Our multi-threaded verification application increases the performance by a factor 3.85, which is an excellent result for a machine with four physical cores.

Operation	Percentage
check PIV	0,2 %
check PAV	1,2 %
Validate Domain pseudonym	5.4 %
Validate Proof	93,7 %
Validate Decryption	0,5 %
check time stamp	0 %

Table 2: Relative amount of work required to validate a single vote in a database with 10 000 votes. Verifying the domain pseudonym requires a database lookup and hence, will grow with an increasing database size.

To illustrate that it is feasible that clients could participate in the verification process, we have verified 1000 votes on our client machine using our multi-threaded application. We took 100 samples, of which the mean value is 74.4 seconds with a standard deviation of 9.4. Taking into account that verifying votes scales almost perfectly linear in time, we can extrapolate that verifying 50 000 votes would take just over 60 minutes on a standard client machine.

6.2 Data consumption

In order to cast a vote, the client first receives the poll description and a 256-bit extended time stamp from the server. The proof that the client returns is a 6 KB XML string, which includes the 0.5 KB large encrypted poll answers. Hence, the amount of data that the client sends to the server is 5.5 KB plus the size of the encrypted answer string. Finally, the server sends a 2048-bit PAV to the user.

As table 1 shows, the database contains a lot of redundant information. The domain pseudonym, time stamp, and encrypted vote columns can all be extracted from the proof. This redundancy, however, does dramatically improve the

performance. Although this redundant information requires the system to store more data, the actual database size is relatively small. One vote record, including decrypted vote, takes up about 20 KB. Hence, a database with 1000 votes is about 20 MB, and a 100 000 vote database contains 2 GB of data.

7. DISCUSSION

This section evaluates the properties of avisPoll. Namely, it discusses security, anonymity, unlinkability, trust and practicality of the design.

7.1 Security

avisPoll implements all necessary mechanisms to protect against attacks with regard to our threat model (Section 4.4).

The system relies on the Idemix anonymous credential system to provide vote verifiability, vote unforgeability and double-voting prevention. A vote is embedded in a valid Idemix proof which satisfies the poll's eligibility policy. In addition, the Idemix proof also proves a domain pseudonym relative to the poll identifier that has not been used in the poll so far. Also, the vote's authenticity is accomplished by signing the actual (encrypted) content of the vote in this proof. Hence, the security of a avisPoll-vote is dependent on the used Idemix security parameters (e.g., a 2048-bit RSA modulus).

The task of the PSP in avisPoll is to gather votes from voters, to validate and to store them. The PSP is not able to influence the poll outcome during this process without being detected. For each vote, the PSP generates a PIV_u, which is a cryptographic hash of the whole vote, and a PAV_u which is the signature of that hash, after which it sends the PAV_u to the voter. If the received PAV_u is valid and the voter's vote does not appear in the final bulletin board, the voter can prove with her PAV_u that the PSP had correctly processed the vote. Hence, she can prove that her vote has to have been omitted from the bulletin board at some point in time. If the PSP returns an invalid PAV_u, or nothing at all, the voter can send a complaint containing her vote (Idemix proof). Using this proof's domain pseudonym, the PO or PA can verify whether or not the vote is present in the bulletin board (once it has been published), and if it is the same vote. However, because the actual answers are encrypted, the PSP cannot make a conscious decision whether or not to to follow the protocol and processing a vote correctly. Hence, the PSP is not able to influence the outcome of the poll. Once the voting phase is completed, the PSP generates a signed Merkle tree, which is verified and signed by the PO and PA. Once this action is completed, nobody is able to modify any poll data without it being detectable using this Merkle tree. Furthermore, each voter is able to verify the existence and integrity of her own vote using her local data (domain pseudonym and PAV_u), and all entities can verify all votes in the bulletin board. Hence, we consider the poll integrity and verifiability requirement as fulfilled.

Finally, avisPoll can minimize the possibility of coercion since the PO can allow voters to update their votes. If voters are coerced to answer in a certain way, they can easily update their vote before the poll is closed. However, this would allow voters to get more than one PAV_u, which they could use to prove the absence of their original vote. However, this problem is solved thanks to the time stamp embedded in the proof. If multiple votes are made under the same domain pseudonym and the votes are valid, then those proofs must have been made by the same voter. A voter thus cannot argue that she did not updated her vote if she is in the possession of the PAV_u from the original vote.

7.2 Anonymity and unlinkability

In addition to many of avisPoll's security properties, its privacy and unlinkability properties are achieved by the use of the Idemix credential system. Note that the anonymity and unlinkability of the communication channel used is orthogonal to our proposed solution and is not discussed in this paper. Although voters are not unlinkable if they perform multiple actions in the same poll, they are unlinkable over multiple polls. Each poll will have a unique poll identifier, and hence, the voter's domain pseudonym will change. This allows avisPoll-credentials to be reused in multiple polls.

Furthermore, the poll's eligibility policy might require voters to reveal information embedded in the credential. This can range from proving that the voter works in the HR department, to proving in a range proof that she is older than eighteen. It is clear that the revealed information should never identify the voter, or put the voter in a small anonymity set.

Because avisPoll uses anonymous credentials, the difficulty of credential revocation arises. However, this can be solved by using a validity period in the avisPoll-credential. For each poll, voters are then required to proof that their credential contains a valid validity period. When the credential expires, the voter is required to update her credential. A similar approach would be to integrate a version number in the credential. Each subsequent poll would specify an increasing version number, making it possible to limit the usage of the credential to maximum N polls by updating the credential's version number with the value M + N (M being the latest poll version number).

7.3 Trust

One of avisPoll's goals is to minimize the required trust assumptions for all entities. In a system setup with two entities, the PSP and PO, the following trust assumptions are required. First, the PO cannot collude with the PSP by sharing the poll specific decryption key. Indeed, if the votes can be decrypted before the Merkle tree is signed and published, the PSP could decrypt votes and discriminate based on its content (i.e. not returning a PAV_u). Second, the PO must only issue one credential per valid voter. Handing out multiple credentials to a particular entity, or to itself, will allow the entity to cast a vote for each credential in its possession, and hence influence the result. However, this attack is limited because the bulletin board containing all the votes is made public as it can be detected if participating in the poll is mandatory, or it is known that only a small part of eligible voters did not vote. Finally, the PO must correctly process incoming complaints (e.g. a missing-vote complaint). avisPoll does not require trust in the PSP except for the first assumption, the fact that the PO and PSP do not collude. We argue that it is not unreasonable that these requirements are no problem for low-risk polls such as electing the representative on a board.

However, if any of these three assumptions cannot be made, avisPoll requires the introduction of one or more additional entities. The first and third assumptions can be met by introducing one or more PAs. The PAs and PO each

hold a share of the decryption key. By requiring N shares to compose the decryption key, avisPoll greatly reduces the possibility that votes can be decrypted before the Merkle tree is signed. Furthermore, the PA does not have a reason to not process complaints correctly, as it does not have interests in the outcome of the poll. Finally, the PAs could audit the PO's credential issuing process to ensure that the right credential is handed to the right person. In addition, avisPoll also allows credentials to be issued by a trusted third party (e.g., the government).

The avisPoll application is very important for avisPoll. It needs to create the votes, verify the results and send complaints. Hence, the application users have to put a lot of trust in this application (e.g., trust that it does not encode an identifier in all communication). For this reason, the complete avisPoll specification to interact with the PSP, PO or PAs should be public knowledge. This means that any party can build its own avisPoll application. Furthermore, making such applications open source is a solution to this trust issue.

7.4 Practicality

Practicality is a very important goal of avisPoll. Because the PSP is a cloud provider, everybody can act as a PO, set up a poll and invite the voters to participate. Voters do require a voting credential in order to participate in a poll. However, a voter can participate in multiple polls using the same credential while access to polls can be limited to a target group.

We also show that avisPoll is scalable and parallellizable. Even voters have the capability to quickly verify a small to medium sized poll, as our results show that it takes just over a minute to verify 1000 votes on a client machine.

Finally, avisPoll relies on voters to check the presence of their own votes. Voters can also validate (part of) the results. This, however, does not benefit the usability of avisPoll. That is why the avisPoll-application should be able to perform these checks automatically, and automatically compose and send complaints.

8. CONCLUSIONS AND FUTURE WORK

This paper described avisPoll, a voting scheme for small to medium-sized, low-risk, Internet-based polls that provides among other features vote anonymity and poll integrity. In addition, the system only requires one entity to collect and process votes, and one entity that organizes the poll and defines the eligible voter set. avisPoll requires its participating entities to only put little trust in each other. However, additional entities might be introduced to lessen these trust requirements to a minimum. A poll service provider can offer its services over the internet. Hence, avisPoll is easy to set up in practice as anybody can use the service provider and organize a poll. Furthermore, our experimental results show that the system is scalable, the processing time does not blow up with regard to any of avisPoll's parameters (e.g. size of poll, number of questions asked, etc.). In addition, vote validation is perfectly parallelizable and can even be done by the voters themselves.

Our future plans for avisPoll is to further develop our prototype implementation into a functional poll service provider and a usable client application for voters and poll organizers. As mentioned in the discussion section, the client application and specification should be open source. This way, voters can fully trust what the application is doing or some might even implement their own version.

9. REFERENCES

[1] B. Adida. Helios: web-based open-audit voting. In *Proceedings of the Usenix Security Symposium*, 2008.

[2] anonymised.

[3] anonymised.

[4] P. Bichsel, J. Camenisch, M. Dubovitskaya, R. R. Enderlein, I. Krontiris, A. Lehmann, G. Neven, J. D. Nielsen, C. Paquin, F.-S. Preiss, et al. H2. 2-abc4trust architecture for developers. *Heartbeat*, 2:2.

[5] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in CryptologyâĂŤEUROCRYPT 2001*, pages 93–118. Springer, 2001.

[6] O. Cetinkaya and A. Doganaksoy. Pseudo-voter identity (pvid) scheme for e-voting protocols. In *Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on*, pages 1190–1196. IEEE, 2007.

[7] M. R. Clarkson, S. Chong, and A. C. Myers. Civitas: Toward a Secure Voting System. In *IEEE Symposium on Security and Privacy*, 2008.

[8] C. Diaz, E. Kosta, H. Dekeyser, M. Kohlweiss, and G. Nigusse. Privacy preserving electronic petitions. *Identity in the Information Society*, 1(1):203–219, 2008.

[9] easypolls. free online polls. https://www.easypolls.net/, 2014.

[10] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, UK, 1987. Springer-Verlag.

[11] A. Holpuch. Taylor swift sends cash to boston deaf school but decides not to play there. http://www.theguardian.com/music/us-news-blog/2012/oct/02/taylor-swift-boston-deaf-school, 2012.

[12] R. Joaquim, A. Zúquete, and P. Ferreira. Revs–a robust electronic voting system. *IADIS International Journal of WWW/Internet*, 1(2):47–63, 2003.

[13] G. Merry. Usrowing board announces results of the 2013 election investigation. http://www.usrowing.org/News/13-05-07/USRowing_Board_Announces_Results_of_the_2013_Election_Investigation.aspx, 2013.

[14] A. E. Project. Patras pilot. https://abc4trust.eu/index.php/home/pilots.

[15] E. Rosenfeld. Mountain dews dub the dew online poll goes horribly wrong. http://newsfeed.time.com/2012/08/14/mountain-dews-dub-the-dew-online-poll-goes-horribly-wrong/, 2012.

[16] SurveyMonkey. https://www.surveymonkey.com/, 2014.

[17] H. Voting. helios: Trust the vote. https://vote.heliosvoting.org/, 2014.